Lab 8

Levon Demirdjian

Tuesday, May 17, 2016

Problem 1: Designing a Metropolis algorithm

Consider a Markov chain over four states 1,2,3,4, and suppose that the target distribution is given by $\pi(1) = 0.2, \pi(2) = 0.2, \pi(3) = 0.3, \pi(4) = 0.3$. We also consider the base chain that moves to one of the other 3 states with probability 1/3 each. The probability of staying in the current state is 0.

1) Write down the transition matrix B of the base chain.

2) Design the Metropolis algorithm. Let M be the transition matrix of the Metropolis algorithm. Write down M and give an interpretation.

Solution:

$$\mathrm{B} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}$$

2)

```
## Step 1: Define the stationary distribution pi and transition matrix B
p <- c(0.2, 0.2, 0.3, 0.3)
B <- rbind(c(0,1/3,1/3,1/3), c(1/3,0,1/3,1/3), c(1/3,1/3,0,1/3), c(1/3,1/3,1/3,0))

## Step 2: Define the Metropolis acceptance step
A <- matrix(nrow = 4, ncol = 4)
for(x in 1:4){
  for(y in 1:4){
    A[x, y] <- min(1, p[y]/p[x]) ## This is P(accept state y | current state is x)
  }
}

## Step 3: Define the transition matrix of the Metropolis algorithm
M <- B * A # Element-wise product
rowSums(M)
```

```
## [1] 1.0000000 1.0000000 0.7777778 0.7777778
```

```
diag(M) <- 1-rowSums(M)
rowSums(M)
```

```
## [1] 1 1 1 1
```

```
M
```

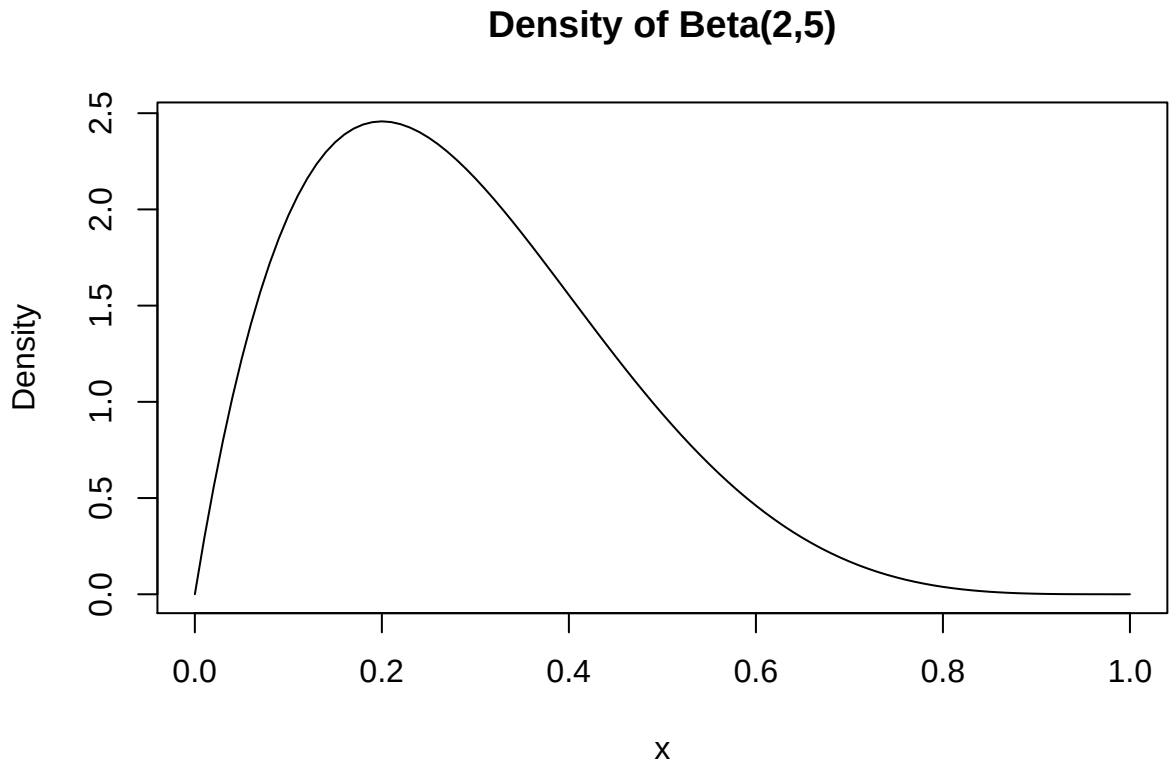
```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.0000000 0.3333333 0.3333333 0.3333333
## [2,] 0.3333333 0.0000000 0.3333333 0.3333333
## [3,] 0.2222222 0.2222222 0.2222222 0.3333333
## [4,] 0.2222222 0.2222222 0.3333333 0.2222222
```

Problem 2: Designing a Metropolis algorithm

In this problem, we are going to design a Metropolis algorithm to sample from the following distribution:

$$\pi(x) \propto x(1-x)^4, \qquad 0 \leq x \leq 1.$$

That is, our target distribution is Beta(2,5). Here is the pdf of this particular beta distribution:

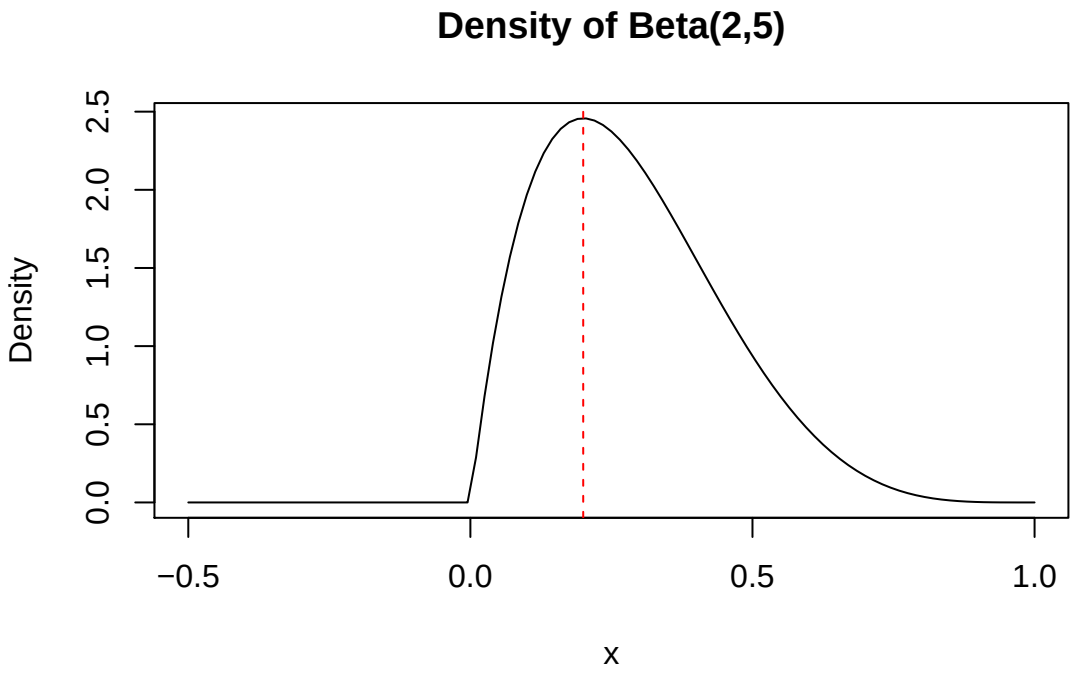


We can think about the metropolis method algorithmically as follows. Say we want to sample from $\pi(x)$ and we have decided to use a normal distribution as our proposal distribution. To sample from $\pi(x)$, we:

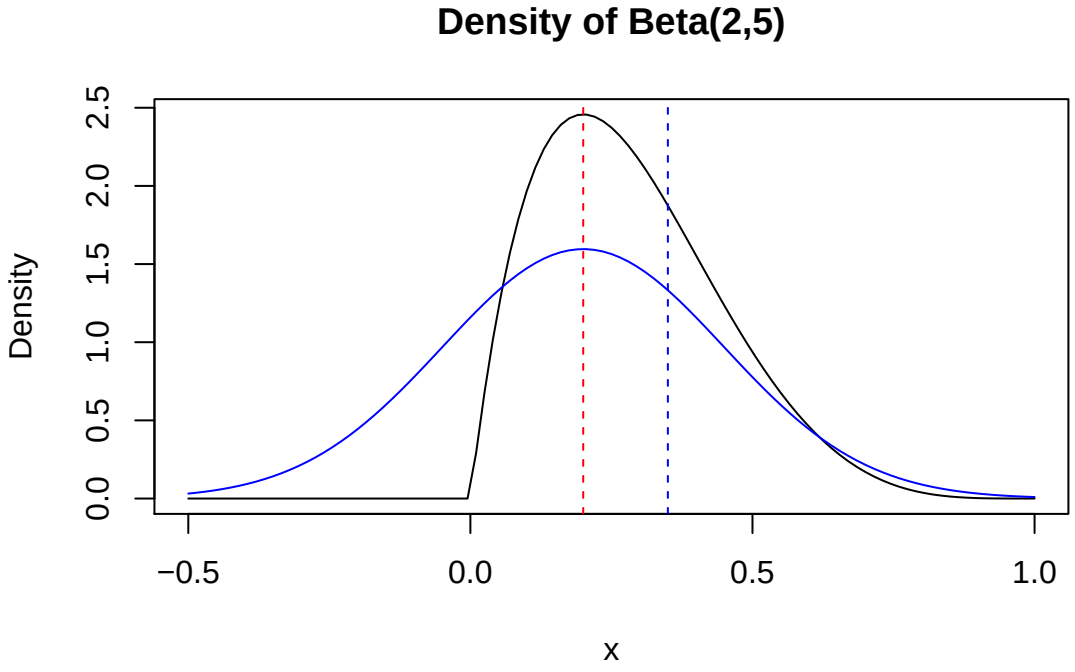
1. Choose a starting point X_0, say $X_0 = 0.2$. We can do this either randomly or systematically.
2. Use the proposal distribution to suggest a new point, X_1.
3. Keep the new point with a certain probability, else keep the old point.

This is illustrated in the next few diagrams when the proposal distribution is a normal distribution:

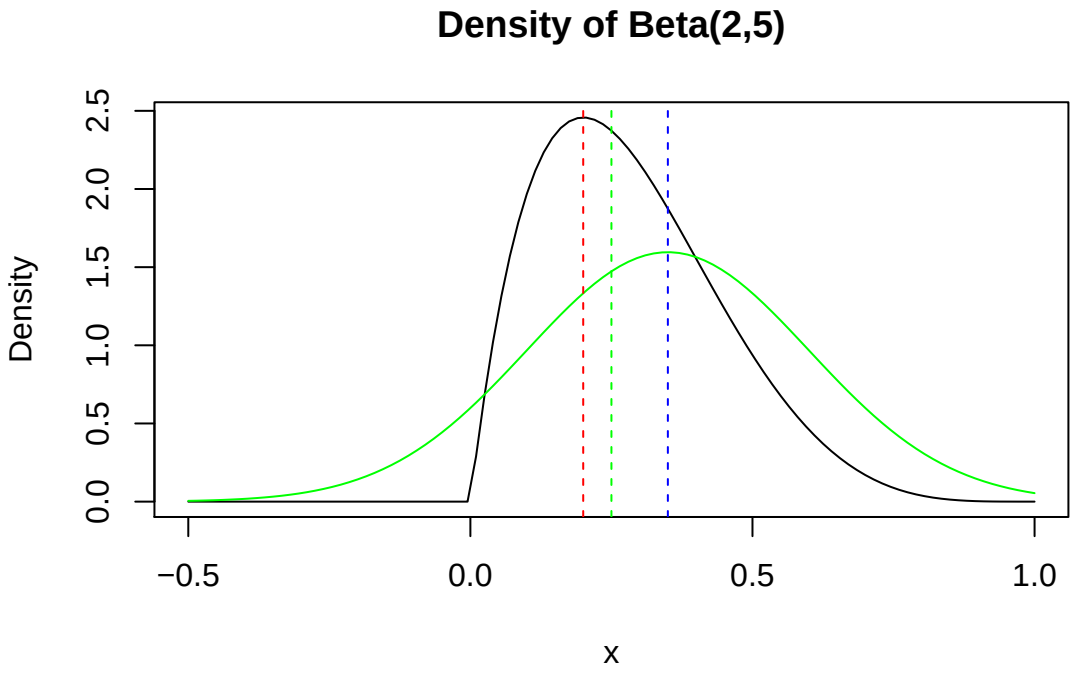
STEP 1: Choose a starting point (red)



STEP 2: Use the proposal distribution to suggest a new point, X_1 (blue)



STEP 3: If accepted, use the proposal distribution to suggest a new point, X_2 (green)



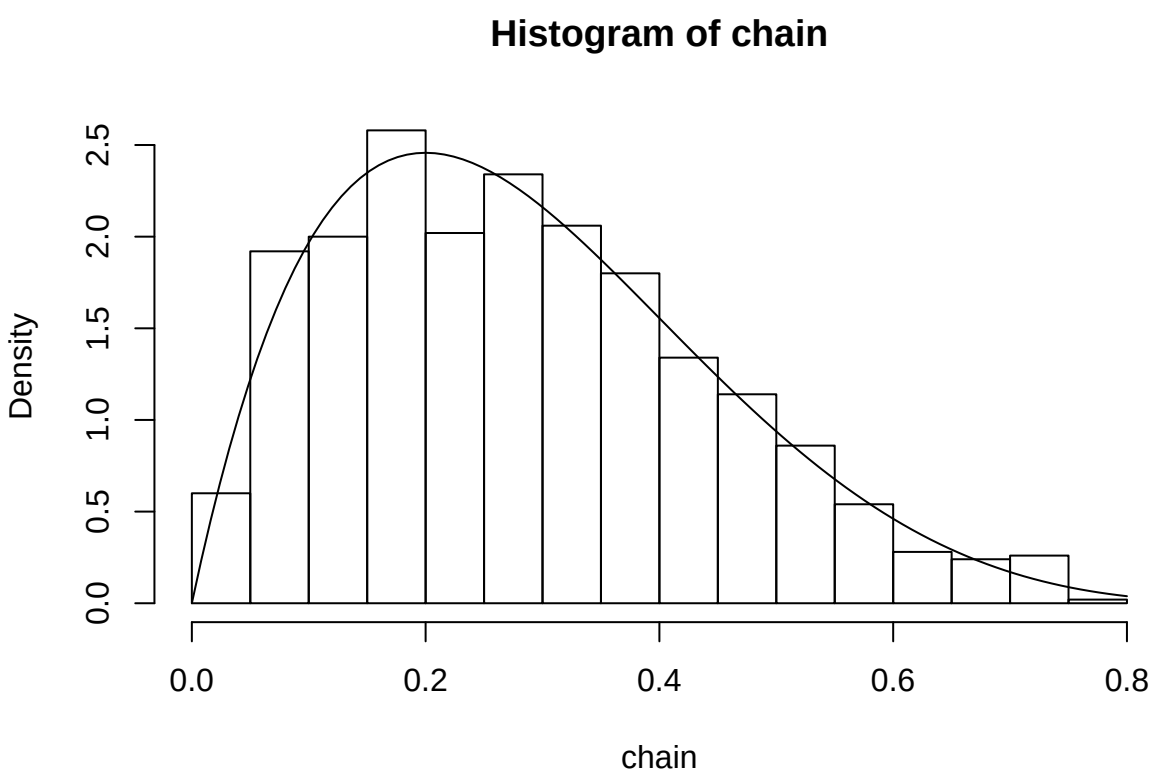
We proceed like this for a large number of iterations until our chain converges. Note that in this example, at each step the proposal distribution (i.e. the normal distribution) is centered over the previous sample..

We will consider three choices for the proposal distribution:

1. The normal distribution
2. The uniform distribution
3. Symmetric random walk

```
# Create a function for the target distribution
f <- function(x) {
  x * (1 - x)^4
}

# Create functions for the proposal distributions
g1 <- function(x_old) {
  x_prop <- rnorm(1, mean = x_old, sd = 0.1)
  x_prop
}

g2 <- function(x_old) {
  x_prop <- runif(1)
  x_prop
}

g3 <- function(x_old) {
```

```
  u <- runif(1)
  ## I'm arbitrarily picking a step size of 0.05 here.
  x_prop <- (x_old + 0.05) * (u > 0.5) + (x_old - 0.05) * (u <= 0.5)
  x_prop
}

n  <- 1000
numSteps <- 100
chain <- rep(0, n)
for(i in 1:n){
  x_old <- 0.2 ## Start at 0.2
  for(j in 1:numSteps) {
    x_prop <- 0
    while(x_prop <= 0 | x_prop >= 1){
      x_prop <- g2(x_old) ## Proposal
    }
    u <- runif(1)
    accept <- u < min(1, f(x_prop) / f(x_old))
    x_old <- x_prop * (accept == 1) + x_old * (accept == 0)
  }
  chain[i] <- x_old
}

## Look at the results
hist(chain, breaks = 20, prob = TRUE)
curve(dbeta(x, 2, 5), add = TRUE)
```


Histogram of chain
Density
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
chain